5-1993

Suboptimal Robot Joint Interpolation Within User-Specified Knot Tolerances

Daniel J. Simon
Cleveland State University, d.j.simon@csuohio.edu

Can Isik
Syracuse University

Publisher's Statement

This is the accepted version of the following article: D. Simon and C. Isik, "Suboptimal Robot Joint Interpolation Within User-Specified Knot Tolerances," J. Robot. Syst., vol. 10, pp. 889-911, 1993., which has been published in final form at http://onlinelibrary.wiley.com/doi/10.1002/rob.4620100702/abstract

Suboptimal Robot Joint Interpolation Within User-Specified Knot Tolerances

Dan Simon
TRW
SB2-1062
PO Box 1310
San Bernardino, CA 92402

Can Isik
121 Link Hall
Department of Electrical Engineering
Syracuse University
Syracuse,NY 13244-1240

1. INTRODUCTION

The industrial robot is a highly nonlinear, coupled multivariable system with nonlinear constraints. For this reason, robot control algorithms are often divided into two stages: *path planning* and *path tracking*. A conceptually simple approach to the path planning problem is to generate a joint-space trajectory based on interpolation of a sequence of desired joint angles. This approach ignores most of the dynamics of the robot, so the resultant trajectories do not take full advantage of the robot's capabilities. But the trajectories are typically computationally inexpensive, making this approach a popular method.[1] In this approach, a number of knot points are chosen along the desired Cartesian path. The number of knots chosen is a trade-off between exactness and computational expense. The Cartesian knots are then mapped into joint knots using inverse kinematics. Finally, an analytic interpolating curve is fit to the joint knots. This curve provides the path tracker with joint angles and derivatives at the controller rate.

The most popular type of interpolation is algebraic splines.[2,3] Lin et al. formulated minimum time[4] and locally based algebraic splines,[5] and Luh and Lin formulated minimum path error algebraic splines.[6] An n^{th}-order algebraic spline consists of piecewise continuous n^{th}-order algebraic polynomials that have continuous derivatives up to order $(n - 2)$ or less (depending on the details of the formulation). Higher-order splines result in continuity of higher-order derivatives. This reduces rough and jerky motions and thus prevents excessive wear on the robot and the excitation of resonances.[1] But, this benefit is at the expense of large oscillations of the trajectory. Polynomials with order as low as five (e.g., quartic splines) can overshoot extreme knots by as much as 60 degrees.[3]

A recent development is the use of trigonometric polynomials to efficiently generate joint trajectories with little overshoot but continuous velocity, acceleration, and jerk.[7,8] Trigonometric polynomials have the characteristic that if they are appropriately normalized in time, they are very smooth,[9] that is, the magnitude of the derivatives are relatively low and the overshoot is relatively small. If piecewise continuous trigonometric polynomials are joined together, the computational expense is low,[10] and each polynomial is of low order, preventing oscillations between knots. These piecewise continuous trigonometric polynomials are called *trigonometric splines*.

In this article, it is shown how a trigonometric spline that passes *near* a given set of joint knots can be optimized. The objective functions that are used are minimum jerk and minimum torque. In the minimum jerk case, the problem reduces to a quadratic programming problem with linear constraints. The maximum error at the knots is specified by the user. The unique contribution of

this article is the straightforward way in which the intermediate knot angle constraints are incorporated into the optimization problem. In addition, the decoupled nature of trigonometric splines can be taken advantage of to reduce the computational expense of the problem.

Section 2 gives a review of trigonometric splines and their application to robot path planning. Section 3 discusses the optimization of trigonometric splines. This includes the case where the trajectory is required to pass exactly through the specified knots and the case where the trajectory is required to pass near the knots within a prespecified tolerance. Section 4 provides some numerical examples of the optimization schemes discussed in this article, and Section 5 presents some concluding remarks.

2. TRIGONOMETRIC SPLINES

The term trigonometric spline was first introduced by Schoenberg,[11] but since then other definitions have appeared in the literature.[12,13] So, the term is not well-defined. In this section, the trigonometric splines used in this article will be defined, and their application to robot path planning will be summarized. See Simon and Isik for details.[7,8]

While Schoenberg was the originator of the term trigonometric spline, his functions are not composed solely of trigonometric functions.[11] Since then, more natural definitions of the term have appeared in the literature.[12,13] The trigonometric splines used in this article are those functions satisfying the following definition.

Definition 1. *An m^{th}-order trigonometric spline function $y(t)$ with a total of $2m$ constraints in each of the n closed arcs $[t_{i-1}, t_i]$ $(i = 1, \ldots, n)$ has the form*

$$y(t) = y_i(t) \quad t \in [t_{i-1}, t_i] \tag{1}$$

where $y_i(t)$ is given by

$$y_i(t) = a_{i,0} + \sum_{k=1}^{m-1} (a_{i,k} \cos kt + b_{i,k} \sin kt) + a_{i,m} \sin m(t - \gamma_i) \tag{2}$$

$$\gamma_i = \sum_{j=0}^{2m-1} \tau_{i,j}/2m \tag{3}$$

and $\tau_{i,j}$ are the values of t where $y_i(t)$ has a constraint applied.

The existence and uniqueness of these functions are asserted by the following theorem.

Theorem 1. *Let $y_i^{(r)}(t)$ denote the rth derivative of $y_i(t)$. If the trigonometric spline functions of Definition 1 satisfy the property that, for each i and j, $y_i^{(r)}(\tau_{i,j})$ is not constrained unless $y_i^{(r-1)}(\tau_{i,j})$ is also constrained $(r = 1, 2, \ldots)$, then the trigonometric spline functions exist and are unique.*

Proof: The proof is long and complicated, and relies heavily on properties appearing in Schoenberg's original article.[11] See Koch and Lyche for a proof.[12,14] ■

A desired continuous time Cartesian trajectory can be discretized into $(n + 1)$ Cartesian goal points at times $t_0 < t_1 < \ldots < t_n$. Then, inverse kinematics can be performed at each of these goal points, resulting in a set of $(n + 1)$ joint space goal points y_i for each joint.

Then, n fourth-order trigonometric polynomials $y_i(t)$ can be generated. Fourth-order polynomials are used so that the first three derivatives at each endpoint can be constrained. This allows the user to join the polynomials together so as to have a joint-space path with continuous derivatives up to the third order. The function $y_i(t)$ $(i = 1, \ldots, n)$ is defined *only* on the time interval $[t_{i-1}, t_i]$. These n trigonometric polynomials are joined together to form a trigonometric spline. Because $y_i(t)$ is a fourth-order trigonometric polynomial, it has eight undetermined coefficients [see eq. (2)]. The eight constraints used to determine the coefficients of $y_i(t)$ are

$$y_i(t_{i-1}) = y_{i-1} \equiv y(t_{i-1})$$

$$y_i(t_i) = y_i \equiv y(t_i)$$

$$y_i^{(r)}(t_{i-1}) = y_{i-1}^{(r)} \equiv y^{(r)}(t_{i-1}) \; (r = 1, 2, 3)$$

$$y_i^{(r)}(t_i) = y_i^{(r)} \equiv y^{(r)}(t_i) \; (r = 1, 2, 3) \tag{4}$$

where $y_i^{(r)}(t)$ denotes the rth derivative of $y_i(t)$. These constraints must be specified (either heuristically or optimally) before the coefficients of $y_i(t)$ can be obtained (see Section 3).

In this article, it will be assumed that $t_{i-1} = 0$ and $t_i = \pi/4$, $(i = 1, \ldots, n)$. These values give computational stability and smoothness of motion.[9] Equation (4) shows that for each spline segment we will have four constraints at $t = 0$ and four constraints at $t = \pi/4$. Therefore, the eight $\tau_{i,j}$s in eq. (3) have the values $(0, 0, 0, 0, \pi/4, \pi/4, \pi/4, \pi/4)$. This results in $\gamma_i = \pi/8$ for all i. Then, eq. (2) becomes the familiar system

$$y_i(t) = a_{i,0} + \sum_{k=1}^{3} (a_{i,k} \cos kt + b_{i,k} \sin kt) + a_{i,4} \cos 4t \quad t \in [0, \pi/4] \tag{5}$$

where the identity $\sin 4(t - \pi/8) = -\cos 4t$ has been used and the sign of $a_{i,4}$ has been reversed for notational simplicity.

The first two constraints of eq. (4) are given by the inverse kinematics solution of the Cartesian trajectory. There are several different ways to specify the last six constraints of eq. (4). One way is that the user may desire certain joint derivatives at the knots. Another possibility is that these constraints could be determined to minimize some objective function (see Section 3). Yet another

possibility is that these constraints could be chosen using some simple, heuristic method (such as a central-difference approximation).[10]

The determination of the eight coefficients for the spline segment $y_i(t)$ can be accomplished by the inversion of an 8×8 matrix (A_i). But this matrix inversion can be performed *a priori*. It does not need to be performed in real time. The 8×8 matrix A_i is a function of only two parameters: t_{i-1} and t_i. So, the time interval of *each* spline segment can be normalized to a *fixed* t_{i-1} and t_i. Then, A_i is a known, constant matrix for all i, and A_i^{-1} is the same for each spline segment. Note that the invertibility of A_i is guaranteed by Theorem 1.

This is the key to the computational benefit of using trigonometric splines.[10] There is *no* need to solve a set of linear equations to determine the spline coefficients. With algebraic splines, the user must know the number of knots on the trajectory before solving the set of linear equations required to determine the spline coefficients. However, when using trigonometric splines, the spline coefficients can be solved by simply multiplying an *a priori* known matrix $(A_i^{-1}$, which is independent of $i)$ by a vector composed of knot angles and derivatives.

Equations (4) and (5) are used to determine the coefficients of the spline segments $y_i(t)$. The multiplication of the 8×8 constant matrix A_i^{-1} by an eight-element vector gives the eight coefficients of $y_i(t)$ as follows:

$$[a_{i,0}\ \ a_{i,1} \cdot \cdot \cdot b_{i,3}\ \ a_{i,4}]^T = A_i^{-1}\,[y_{i-1}\ \ y_i\ \ y'_{i-1} \cdot \cdot \cdot \ y'''_{i-1}\ \ y'''_i]^T \tag{6}$$

Note that the invertibility of A_i is guaranteed by Theorem 1. See Simon and Isik for the numerical value of the A_i matrix.[7,8] Because the segment $y_i(t)$ is defined on $t \in [0,\pi/4]$ for all i, the time-scaled trigonometric spline $y(t)$ is given by

$$y(t + (i - 1)\pi/4) = y_i(t), \quad t \in [0,\pi/4], \quad (i = 1, \ldots, n) \tag{7}$$

The function $y(t)$ is a trigonometric spline that satisfies the desired interpolation conditions and has length $n\pi/4$ s. The unscaled spline $\theta(t)$ given by

$$\theta(t) = y(n\pi t/4T) \quad t \in [0,T] \tag{8}$$

stretches the trajectory from its normalized length $n\pi/4$ to a desired length T. The derivatives of the unscaled trajectory are related to the derivatives of the scaled trajectory as follows:

$$\theta^{(r)}(t) = (n\pi/4T)^r y^{(r)}(n\pi t/4T) \tag{9}$$

3. OPTIMIZATION

The user of the trajectory formulation algorithm described in the previous section is free to choose the first three trajectory derivatives at each knot. The user will typically desire to set the derivatives at the endpoints to zero. A

simple and reasonable heuristic method of choosing the first derivative at the interior knots would be to use a central-difference method on the knot angles.[10] Similarly, a difference method could be used on the knot velocities to calculate the interior knot accelerations, and a difference method could be used on the knot accelerations to calculate the interior knot jerks. The resultant trajectories are called *nominal* trigonometric splines.

However, if additional computer time is available, the knot derivatives can be chosen to minimize some objective function. A general objective function can be written in the form

$$J = f[\vec{\theta}(t)] \tag{10}$$

where $f(\cdot)$ is a general nonlinear function, $\vec{\theta}(t)$ is a P-vector of trigonometric splines, and P is the number of joints that the robot has.

Recall that $\vec{\theta}(t)$ is a time-scaled and time-shifted version of $\vec{y}(t)$ [see eq. (8)] where $\vec{y}(t)$ is the P-vector of normalized trigonometric splines, and $\vec{y}(t)$ is composed of the n spline segments $\vec{y}_i(t)$ [see eq. (7)]. Therefore, eq. (10) can be written as

$$J = g\left[\sum_{k=1}^{n} \vec{y}_k(t)\right] \tag{11}$$

where $g(\cdot)$ is a general nonlinear function of the same form as $f(\cdot)$ in eq. (10) but with the inclusion of appropriate time-scaling constants. The minimization of this general objective function becomes a parameter optimization problem because $\vec{y}(t)$ is a function of the $3P(n-1)$ free knot derivatives, where $(n-1)$ is the number of interior knots of each joint trajectory.

If J is known to have only one minimum, then eq. (11) is minimized when

$$\frac{\partial}{\partial y_{i,j}^{(r)}} g\left[\sum_{k=1}^{n} \vec{y}_k(t)\right] = 0, \quad (i = 1, \ldots, n-1), (r = 1, 2, 3), \quad (j = 1, \ldots, P) \tag{12}$$

where $y_{i,j}^{(r)}$ is the r^{th} derivative of the i^{th} normalized spline segment of the j^{th} joint of the robot. Because $\vec{y}_k(t)$ is a function of $y_{i,j}^{(r)}$ only for $k \in \{i, i+1\}$ [see eqs. (5) and (6)], eq. (12) can be written as

$$\frac{\partial}{\partial y_{i,j}^{(r)}} g[\vec{y}_i(t) + \vec{y}_{i-1}(t)] = 0, (i = 1, \ldots, n-1), (r = 1,2,3), (j = 1, \ldots, P) \tag{13}$$

Further simplification from this point depends on the form of eq. (10).

So, the optimal control problem is simplified by reducing its dimension, thereby converting it to a parameter optimization problem. This general approach to optimal control is similar to that taken by others.[15–21] But, their formulations are applicable only for constraints at the initial and final time and

do not allow for constraints at specific times in between. In other words, as applied to the robot path planning problem, their approaches are valid only for path planning between an initial point and a final point, and do not allow for intermediate knots.

Two specific examples of optimization (minimum jerk and minimum torque) are considered in the following sections.

3.1. Minimum Jerk Trajectory

Suppose that the user desires to minimize the jerk of each joint throughout its trajectory. Kyriakopoulos and Saridis report that the joint position errors of the path tracker increase with the magnitude of joint jerk.[22] Also, Flanagan and Ostry present evidence that the human brain plans arm movements so as to minimize a function of joint jerk.[23] So, minimizing some function of joint jerk would seem to be desirable, resulting in a coordinated motion that could be accurately followed by the robot path tracker. The objective function of eq. (10) could then be written as

$$J = \int_0^T [\theta'''(t)]^2 \mathrm{d}t \tag{14}$$

Because the jerk of each joint is decoupled from the other joints, the minimization of eq. (14) can be performed one joint at a time. Because $\theta'''(t)$ is a scaled version of $y'''(t)$, the minimization of eq. (14) is equivalent to the minimization of

$$J = \int_0^{n\pi/4} [y'''(t)]^2 \mathrm{d}t \tag{15}$$

where $(n + 1)$ is the number of knots and $n\pi/4$ is the normalized length of the joint trajectory [see eq. (7)]. To minimize eq. (15), we want to set each of the partial derivatives with respect to the $(n - 1)$ normalized interior knot derivatives equal to zero. So, eq. (15) will be minimized when

$$\frac{\partial}{\partial y_i^{(r)}} \left\{ \int_0^{n\pi/4} [y'''(t)]^2 \mathrm{d}t \right\} = 0 \quad (i = 1, \ldots, n-1), \quad (r = 1, 2, 3) \tag{16}$$

Recall that $y(t)$ is composed of the functions $y_i(t)$ $(i = 1, \ldots, n)$, each of which is an analytic function of the eight parameters $(y_j^{(r)})$, $(j = i - 1, i)$, $(r = 0, 1, 2, 3)$ [see eqs. (5) and (6)]. So, the differentiation and integration of eqs. (13) and (15) can be performed analytically to obtain the $(n - 1)$ matrix equations

$$D_1 \begin{pmatrix} y_{i-1} \\ y_i \\ y_{i+1} \end{pmatrix} + D_2 \begin{pmatrix} y'_{i-1} \\ y''_{i-1} \\ y'''_{i-1} \end{pmatrix} + D_3 \begin{pmatrix} y'_i \\ y''_i \\ y'''_i \end{pmatrix} + D_4 \begin{pmatrix} y'_{i+1} \\ y''_{i+1} \\ y'''_{i+1} \end{pmatrix} = 0, \quad (i = 1, \ldots, n-1) \tag{17}$$

where the D_k are 3×3 matrices. Assume that the derivatives of the trajectory are constrained at the endpoints, and adopt the notation

$$Y_i^{(r)} \equiv (y_i' \quad y_i'' \quad y_i''')^T \tag{18}$$

Then, eq. (17) can be combined into the block tridiagonal matrix equation

$$\begin{pmatrix} D_3 & D_4 & 0 & 0 & 0 & \cdots & 0 \\ D_2 & D_3 & D_4 & 0 & 0 & \cdots & 0 \\ 0 & D_2 & D_3 & D_4 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & D_2 & D_3 & D_4 \\ 0 & \cdots & \cdots & \cdots & 0 & D_2 & D_3 \end{pmatrix} \begin{pmatrix} Y_1^{(r)} \\ Y_2^{(r)} \\ Y_3^{(r)} \\ \vdots \\ Y_{n-2}^{(r)} \\ Y_{n-1}^{(r)} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \\ \vdots \\ C_{n-2} \\ C_{n-1} \end{pmatrix} \tag{19}$$

where the 3×3 D_k matrices are constant matrices with known numerical entries and the 3×1 C_k vectors are constant vectors with known numerical entries. See Simon and Isik for details.[7,8] It can be shown that the matrix on the left-hand side of eq. (19) is always nonsingular. This property follows from the fact that eqs. (14) and (15) are always greater than zero unless all of the knot derivatives are zero.

3.2. Minimum Torque Trajectory

Recall that the P-element torque vector of a P-joint robot can be given as

$$\vec{\mathcal{T}}(\vec{\theta}) = M(\vec{\theta})\vec{\theta}'' + S(\vec{\theta}, \vec{\theta}') \tag{20}$$

where M is the $P \times P$ mass matrix and S is a vector of centrifugal, Coriolis, and gravity terms. Suppose the user wants to choose the interior knot derivatives of the trigonometric spline for each joint so as to achieve a minimum torque trajectory. Then, the objective function could be written as

$$J = \int_0^T \vec{\mathcal{T}}^T(\vec{\theta}) R \vec{\mathcal{T}}(\vec{\theta}) \mathrm{d}\tau \tag{21}$$

where R is a $P \times P$ positive-definite weighting matrix. Using the fact that $\vec{\theta}(\tau) = \vec{y}(n\pi\tau/4T)$ [see eq. (8)], the torque vector $\vec{\mathcal{T}}$ can be written as the following function of the normalized joint trajectories:

$$\begin{aligned} &\vec{\mathcal{T}}(\vec{\theta}(\tau)) \\ &\quad = M(\vec{y}(n\pi\tau/4T))(n\pi/4T)^2\ \vec{y}''(n\pi\tau/4T) + S(\vec{y}(n\pi\tau/4T), (n\pi/4T)\vec{y}'(n\pi\tau/4T)) \end{aligned} \tag{22}$$

Using the change of variables $t = n\pi\tau/4T$, the objective function of eq. (21) can be written in terms of the normalized joint trajectories as

$$J = \frac{4T}{n\pi}\int_0^{n\pi/4} \mathcal{T}_y^T R \mathcal{T}_y \mathrm{d}t \tag{23}$$

where $\mathcal{T}_y$ is given by

$$\mathcal{T}_y = M(\vec{y})(n\pi/4T)^2\vec{y}'' + S(\vec{y},(n\pi/4T)\vec{y}') \tag{24}$$

Because $\vec{y}$ is formed by joining together the individual $\vec{y}_i$ components, each of which is defined only on the time interval $t \in [0, \pi/4]$ [see eq. (7)], we obtain

$$J = \frac{4T}{n\pi}\int_0^{\pi/4} \sum_{i=1}^{n} \mathcal{T}_{yi}^T R \mathcal{T}_{yi} dt \equiv \sum_{i=1}^{n} J_i \tag{25}$$

where $\mathcal{T}_{yi}$ is the normalized torque vector that is applied during the i^{th} spline segment, that is, $\mathcal{T}_{yi}$ is equal to eq. (24) when $\vec{y}$ is replaced by $\vec{y}_i$.

So J is completely determined by the $3P(n-1)$ free parameters $\vec{y}_i^{(r)}$, $(r = 1, 2, 3)$, $(i = 1, \ldots, n-1)$. The optimal control problem of eq. (21) has thus been converted into a parameter optimization problem. Note that the objective function could also be minimized with respect to T (the actual path length) using some parameter optimization scheme.

A significant computational savings in the solution of eq. (25) can be realized by taking advantage of the fact that $\mathcal{T}_{yj}$ is an explicit function of $\vec{y}_i^{(r)}$ only for $j \in \{i, i+1\}$. This fact is due to the decoupling of the spline segments. Therefore, eq. (25) can be solved by solving the following $(n-1)$ problems:

$$\min_{\vec{y}_i^{(r)}}(J_i + J_{i+1}) \; (i = 1, \ldots, n-1) \tag{26}$$

where J_i is defined in eq. (25). So, the $3P(n-1)$-dimensional minimization problem of eq. (25) has been converted into $(n-1)$ separate minimization problems, each of dimension $3P$. Of course, eq. (26) is a highly nonlinear function of the parameters $\vec{y}_i^{(r)}$. Some numerical method can be used to find the minimum of these functions and thus to determine the interior knot derivatives that yield a minimum torque trigonometric spline.

For example, Powell's method can be used to find a minimum of a multidimensional function J.[24] Powell's method consists of a sequence of line minimizations, and does not require any derivative calculations. Each line minimization minimizes the function along one direction. The line minimization is accomplished by first finding three points (a, b, c) along the given line such that $a < b < c$ and $J(b) < J(a)$ and $J(b) < J(c)$. Then, these three points are used to interpolate a quadratic function. The minimization of this quadratic function is easily found analytically, giving a new point b'. Then, the pair $(b', J(b'))$ replaces one of the three old points, and the quadratic minimization is repeated.

This process continues until convergence is attained, at which point the function J has been minimized along one direction.

Powell's method repeats the above line minimization N times, where N is the dimension of the domain of the function J. Each set of N line minimizations is called an *iteration*. After each iteration, one of the N directions is replaced by a new direction to speed convergence. So the next iteration begins, which again minimizes along N directions. One of those directions is new,and $(N - 1)$ of the directions are the same as those used in the previous iteration. The new direction is the average direction moved in the previous iteration. The direction that resulted in the largest decrease of the objective function is the direction that is replaced.

3.3. Optimization with Nonzero Knot Tolerances

As described in Section 2, inverse kinematics are used at a sequence of desired robot configurations to obtain a sequence of joint angles. These joint angles are interpolated by a trigonometric spline. The resultant spline will exactly give the desired robot configurations at the knots. However, the knots are often chosen to avoid obstacles or satisfy joint angle limit constraints. So the user may not really require the robot trajectory to pass exactly through the knots. It is possible that the knots are more like "centers of tolerance" *near* which the robot is required to pass. This is the trajectory planning approach used by Paul.[25] Unfortunately, Paul's method does not result in any *a priori* error bounds at the knots.

The trigonometric splines discussed earlier in this article, and most algebraic splines, are planned so as to exactly pass through the given knots. The remainder of this section discusses the use of trigonometric splines when the robot is not required to pass exactly through the given knots. This additional freedom is used to improve the performance of the robot trajectory with respect to the objective functions discussed earlier in this section: minimum jerk and minimum torque.

Typically, we would expect knot tolerances to be given in task space. These tolerances must be mapped into joint space to perform the constrained optimization discussed in the remainder of this section. In this article, it is assumed that the knot tolerances have indeed been mapped into joint space.

3.3.1. *Minimum Jerk Trajectory with Nonzero Knot Tolerances*

As before, we desire to minimize the integral of the square of the jerk of the joint trajectory

$$\int_0^T [\theta'''(t)]^2 \mathrm{d}t \tag{27}$$

As seen earlier (see Section 3.1), this is equivalent to the problem of minimizing

$$\int_0^{n\pi/4} [y'''(t)]^2 \mathrm{d}t \tag{28}$$

where $y(t)$ is the normalized trigonometric spline, and this problem is equivalent to the minimization of

$$\sum_{i=1}^{n} \int_0^{\pi/4} [y_i'''(t)]^2 \mathrm{d}t \tag{29}$$

where $y_i(t)$ is the i^{th} normalized trigonometric spline segment. As described in Section 3.1, $y_i'''(t)$ is a linear function of the eight parameters $y_j^{(r)}$, $(j = i - 1, i)$, $(r = 0, 1, 2, 3)$. Therefore

$$\int_0^{\pi/4} [y_i'''(t)]^2 \mathrm{d}t = \frac{1}{2} x_i^T \hat{Q} x_i \tag{30}$$

where x_i^T is given by

$$x_i^T = (y_{i-1} \quad y_{i-1}' \quad . \quad . \quad . \quad y_i''') \tag{31}$$

and $\hat{Q}$ is an 8×8 symmetric positive semidefinite matrix. So, we obtain

$$\int_0^{n\pi/4} [y'''(t)]^2 \mathrm{d}t = \frac{1}{2} \sum_{i=1}^{n} x_i^T \hat{Q} x_i \tag{32}$$

Now, partition the vectors x_i and the matrix $\hat{Q}$ as

$$x_i = (\phi_{i-1}^T \quad \phi_i^T)^T \tag{33}$$

$$\hat{Q} = \begin{pmatrix} \hat{Q}_{11} & \hat{Q}_{12} \\ \hat{Q}_{12}^T & \hat{Q}_{22} \end{pmatrix} \tag{34}$$

where each submatrix in $\hat{Q}$ is a 4×4 matrix, and

$$\phi_i^T = (y_i \quad y_i' \quad y_i'' \quad y_i''') \tag{35}$$

Using this partition, we obtain

$$\begin{aligned} &\frac{1}{2} \sum_{i=1}^{n} x_i^T \hat{Q} x_i \\ &= \frac{1}{2} \sum_{i=1}^{n} (\phi_{i-1}^T \hat{Q}_{11} \phi_{i-1} + 2\phi_{i-1}^T \hat{Q}_{12} \phi_i + \phi_i^T \hat{Q}_{22} \phi_i) \\ &= \frac{1}{2} [\phi_0^T \hat{Q}_{11} \phi_0 + \phi_n^T \hat{Q}_{22} \phi_n + 2\phi_0^T \hat{Q}_{12} \phi_1 + 2\phi_{n-1}^T \hat{Q}_{12} \phi_n + \\ &\quad 2(\phi_1^T \, . \, . \, . \, \phi_{n-2}^T) \operatorname{diag}(\hat{Q}_{12})(\phi_2^T \, . \, . \, . \, \phi_{n-1}^T)^T + \\ &\quad (\phi_1^T \, . \, . \, . \, \phi_{n-1}^T) \operatorname{diag}(\hat{Q}_{11} + \hat{Q}_{22})(\phi_1^T \, . \, . \, . \, \phi_{n-1}^T)^T] \end{aligned} \tag{36}$$

where diag($\hat{Q}_{12}$) is a $4(n-2) \times 4(n-2)$ block diagonal matrix with $\hat{Q}_{12}$ being the 4×4 matrix located at each diagonal position. A similar description holds for the $4(n-1) \times 4(n-1)$ matrix diag($\hat{Q}_{11} + \hat{Q}_{22}$). Further manipulations yield the equation

$$\frac{1}{2}\sum_{i=1}^{n} x_i^T \hat{Q} x_i =$$

$$\frac{1}{2}\phi_0^T \hat{Q}_{11}\phi_0 + \frac{1}{2}\phi_n^T \hat{Q}_{22}\phi_n +$$

$$(\phi_0^T \hat{Q}_{12}\, 0 \ldots 0\, \phi_n^T \hat{Q}_{12}^T)(\phi_1^T \ldots \phi_{n-1}^T)^T +$$

$$\frac{1}{2}x^T \begin{pmatrix} \hat{Q}_{11}+\hat{Q}_{22} & \hat{Q}_{12} & 0 & \ldots & 0 \\ \hat{Q}_{12}^T & \hat{Q}_{11}+\hat{Q}_{22} & \hat{Q}_{12} & \ldots & 0 \\ 0 & \hat{Q}_{12}^T & \hat{Q}_{11}+\hat{Q}_{22} & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & \hat{Q}_{11}+\hat{Q}_{22} & \hat{Q}_{12} \\ 0 & \ldots & \ldots & \hat{Q}_{12}^T & \hat{Q}_{11}+\hat{Q}_{22} \end{pmatrix} x \tag{37}$$

where the $4(n-1)$-vector x is given by

$$x^T = (\phi_1^T \ldots \phi_{n-1}^T) \tag{38}$$

Equation (37) is the quantity that we are trying to minimize. But, the first two terms on the right-hand side of eq. (37) are constants. So, we can write the minimization problem as

$$\min \int_0^T [\theta'''(t)]^2 dt = \min_x \left(\frac{1}{2} x^T Q x - b^T x \right) \tag{39}$$

where the $4(n-1)$-vector b is given by

$$b^T = -(\phi_0^T \hat{Q}_{12} \quad 0 \ldots 0 \quad \phi_n^T \hat{Q}_{12}^T) \tag{40}$$

and the block tridiagonal matrix Q is obvious from eq. (37). The $\hat{Q}$ matrices of eq. (34) and following are given numerically in Simon.[9]

Now, the user may not require the trigonometric spline to pass exactly through the given interior knots. The user may rather specify a desired tolerance for each knot. This increases the domain of the optimization problem and thus results in a lower objective function value and a larger computational effort. The knot tolerances result in the following inequality constraints being associated with the minimization problem of eq. (27).

$$|y_i - y_{ci}| \leq y_{toli} \quad (i = 1, \ldots, n-1) \tag{41}$$

where y_i is the angle of the trigonometric spline at knot i, y_{ci} is the desired knot angle (the center of tolerance), and y_{toli} is the allowable joint angle error at knot i. These constraints can be written as

$$Ax \leq c \tag{42}$$

where A is the $2(n-1) \times 4(n-1)$ matrix

$$A = \begin{pmatrix} A_s & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A_s \end{pmatrix} \tag{43}$$

and

$$A_s = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{44}$$

The vector x is the $4(n-1)$-vector given in eq. (38), and c is the $2(n-1)$-vector given by

$$c^T = [(y_{tol1} - y_{c1})\,(y_{tol1} + y_{c1})\,(y_{tol2} - y_{c2}) \cdots (y_{tol(n-1)} + y_{c(n-1)})] \tag{45}$$

So, by combining eq. (39) and (42), the minimum jerk problem with nonzero knot tolerances can be written as

$$\min_x \left(\frac{1}{2} x^T Q x - b^T x \right) \quad \text{subject to } Ax \leq c \tag{46}$$

Matrix Q can be shown to be positive definite by similar reasoning as used for the matrix in eq. (19).

So, the problem has been reduced to a quadratic programming problem with linear constraints. This type of problem can be solved by several different algorithms, among which is the following.[26,27]

3.3.1.1. *Hildreth's Algorithm.* A locally minimum solution vector x_0 to the problem of eq. (46) can be found by setting

$$x_0 = Q^{-1}(b - A^T \lambda_0) \tag{47}$$

where λ_0 is the solution to the dual problem

$$\min_\lambda \left(\frac{1}{2} \lambda^T P x + \lambda^T d \right) \text{ subject to } \lambda \geq 0 \tag{48}$$

and P and d are given by

$$P = AQ^{-1}A^T \tag{49}$$

$$d = c - AQ^{-1}b \tag{50}$$

Use the notation that λ_i^k is the i^{th} component of the vector λ at the k^{th} iteration, and let N be the number of components of vectors λ and x. Then, eq. (48) can be solved by the following iterative method:

$$\lambda_i^{k+1} = \max\left[0, -\frac{1}{p_{ii}}\left(d_i + \sum_{j=1}^{i-1} p_{ij}\lambda_j^{k+1} + \sum_{j=i+1}^{N} p_{ij}\lambda_j^k\right)\right] \tag{51}$$

Equation (51) is first performed for $i = 1, \ldots, N$. Then, k is increased by one and the next iteration begins. This algorithm is developed by Hildreth and summarized by Luenberger.[26,27] Note that trajectory derivative inequality constraints at the knots can easily be incorporated into this problem by a straightforward modification of matrix A [eq. (42)].

3.3.2. *Minimum Torque Trajectory with Nonzero Knot Tolerances*

Section 3.2 discussed the choice of the interior knot derivatives that would result in a minimum torque trigonometric spline. The optimum knot derivatives were found by using Powell's method, an iterative numerical minimization procedure.

Now suppose we desire to find a minimum torque trigonometric spline through a given sequence of knots, but with a specified allowable knot tolerance [see eq. (41)] given by

$$|\vec{y}_i - \vec{y}_{ci}| \leq \vec{y}_{toli} \quad (i = 1, \ldots, n-1) \tag{52}$$

Each vector in eq. (52) has P elements, with P being the number of joints of the robot. The vector inequality in eq. (52) is taken component by component.

In principle, Powell's method can be used to find a minimum torque trigonometric spline subject to the constraints given by eq. (52). This is done by simply augmenting the torque objective function with a penalty function.[27] For instance, the constrained problem

$$\min[f(x)] \quad \text{subject to} \quad g_i(x) \leq 0 \quad (i = 1, \ldots, m) \tag{53}$$

is equivalent to the unconstrained problem

$$\min\left\{f(x) + \sum_{i=1}^{m} k_i g_i^2(x) u[g(x)]\right\} \tag{54}$$

where $u(\cdot)$ is the unit step function, and k_i are weighting constants. Unfortunately, when $f(x)$ is a highly coupled nonlinear function, the penalty function approach may result in an augmented objective function with many hills and valleys. So a local minimum of eq. (54) might be significantly larger than other nearby minima. This indeed turns out to be the case for the minimum torque trigonometric spline problem with nonzero knot tolerances. Specifically, consider the constrained minimization problem

$$\min \int_0^T \mathcal{T}^T R \mathcal{T} \, d\tau \quad \text{subject to } |\vec{y}_i - \vec{y}_{ci}| \leq \vec{y}_{toli} \quad (i = 1, \ldots, n-1) \tag{55}$$

This problem can be converted into an unconstrained problem using the penalty function approach and then solved by Powell's method. But the solution thus obtained is neglibly better than the solution to the problem

$$\min \int_0^T \mathcal{T}^T R \mathcal{T} \, d\tau \quad \text{subject to } \vec{y}_i = \vec{y}_{ci} \quad (i = 1, \ldots, n-1) \tag{56}$$

which is simply the unconstrained minimum torque problem with zero knot tolerances (see Section 3.2). So, rather than using a penalty function method the following method, which makes use of the physical significance of the constraints of eq. (52), is proposed:

$$\min_{\vec{y}_i} \left[\min_{\vec{y}_i^{(r)}} \int_0^{\pi/4} (\mathcal{T}_i^T R \mathcal{T}_i + \mathcal{T}_{i+1}^T R \mathcal{T}_{i+1}) d\tau \right], \quad (i = 1, \ldots, n-1), \quad (r = 1, 2, 3) \tag{57}$$

This method is a set of $(n-1)$ minimizations (the outer minimization) over P-dimensional domains $(\vec{y}_i)$. The function that each of these $(n-1)$ minimizations minimizes is itself the solution to a minimization problem (the inner minimization) over $3P$-dimensional domains $(\vec{y}_i^{(r)})$.

The above algorithm recognizes the increased number of hills and valleys in the objective funtion due to the increased number of free parameters (i.e., the knot angles). The algorithm also recognizes that the optimum knot derivatives $\vec{y}_i^{(r)}$ are functions of the knot angles $\vec{y}_i$.

3.4. Summary of Optimization Results

Optimization of trigonometric splines has been discussed in this section. A trigonometric spline trajectory of a P-joint robot is a function of $3P(n-1)$ parameters, where $(n-1)$ is the number of interior knots. The free parameters are the first three derivatives of each joint at each of the interior knots. So a robot trajectory planning problem is converted to a parameter optimization problem. Constraints on intermediate robot configurations are realized by adding an appropriate knot to the trigonometric spline.

If the objective function is an arbitrary combination of joint derivatives, and the constraints on the intermediate robot configurations are equality constraints, then the optimization problem has a unique solution that can be solved in closed form. If the constraints are inequality constraints, or if a general objective function (e.g., torque) is used, the problem must be solved using an iterative method. This is due to the nonlinearity and coupling of robot dynamics.

There is nothing new about assuming the form of the control and thus converting the optimal control problem into a parameter optimization problem. The unique contribution of this section is the straightforward way in which intermediate constraints can be incorporated into the problem. A less significant but equally unique contribution is the decoupled nature of the trigonometric spline segments, which results in a large savings of computational effort for the iterative minimization methods (see Section 3.2).

4. SIMULATION RESULTS

In this section, simulation results will be presented to support the work done in the previous section. The robot manipulator that is considered is a two-link robot that is described in Craig.[1] A two-degree-of-freedom robot is used so that the results can be easily shown in figures. The robot operates in the vertical plane with the acceleration due to gravity denoted by g. It is assumed that each link's mass (m_1 and m_2) is concentrated at its distal end. The link lengths are denoted by l_1 and l_2. The torque (in Newton meters) due to viscous friction for each joint is assumed to be five times the joint velocity (in rad/s). The first joint angle θ_1 is taken as the angle from the horizontal positive x-direction to the first link. The second joint angle θ_2 is taken as the angle from the outward direction of the first link to the second link. Both joint angles are measured in the counterclockwise direction. The joint torques for this simple manipulator are given by

$$\tau_1 = m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + m_2 l_1 l_2 c_2(2\ddot{\theta}_1 + \ddot{\theta}_2) + (m_1 + m_2) l_1^2 \ddot{\theta}_1 - m_2 l_1 l_2 s_2 \dot{\theta}_2^2 - 2m_2 l_1 l_2 s_2 \dot{\theta}_1 \dot{\theta}_2 + m_2 l_2 g c_{12} + (m_1 + m_2) l_1 g c_1 + 5\dot{\theta}_1 \tag{58}$$

$$\tau_2 = m_2 l_1 l_2 c_2 \ddot{\theta}_1 + m_2 l_1 l_2 s_2 \dot{\theta}_1^2 + m_2 l_2 g c_{12} + m_2 l_2^2(\ddot{\theta}_1 + \ddot{\theta}_2) + 5\dot{\theta}_2 \tag{59}$$

where the shorthand notation c_i means $\cos(\theta_i)$, c_{ij} means $\cos(\theta_i + \theta_j)$, and similarly for s_i. The robot parameters are taken to be

$$l_1 = l_2 = 0.5 \text{ m} \tag{60}$$

$$g = 9.8 \text{ m/s}^2 \tag{61}$$

$$m_1 = 4.6 \text{ kg} \tag{62}$$

$$m_2 = 2.3 \text{ kg} \tag{63}$$

Table I. Seven Cartesian and joint space knots.

	Cartesian knots (ms)		Joint angles	
Knot number	x	y	1	2
1	$\sqrt{2}/2$	0	45	−90
2	$\sqrt{2}/2$	$\sqrt{2}/4$	64	−76
3	$\sqrt{2}/4$	$\sqrt{2}/2$	101	−76
4	0	1	90	0
5	$-\sqrt{2}/4$	$\sqrt{2}/2$	79	76
6	$-\sqrt{2}/2$	$\sqrt{2}/4$	116	76
7	$\sqrt{2}/2$	0	135	90

Seven Cartesian knots are specified. The trigonometric spline is required to pass through (or near in the case of nonzero knot tolerances) these seven knots. The seven knots are given in Table I, along with the corresponding joint angles at the knots (obtained by inverse kinematics). The seven knots are shown graphically in Figure 1. There is currently no other literature that discusses optimum robot path planning through a given set of knots. So, these knots were chosen somewhat arbitrarily to represent what might be a typical task for an industrial robot. The length of the path was fixed at 30 s. In this section, five different types of trigonometric splines are computed:

- nominal splines (no optimization)
- minimum jerk splines (Section 3.1)
- minimum torque splines (Section 3.2)
- minimum jerk splines with nonzero (4°) knot tolerances (Section 3.3.1)
- minimum torque splines with nonzero (4°) knot tolerances (Section 3.3.2).

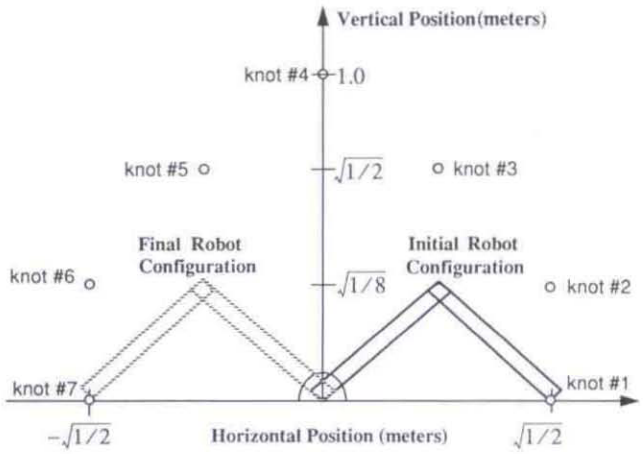

Figure 1. Seven Cartesian knots.

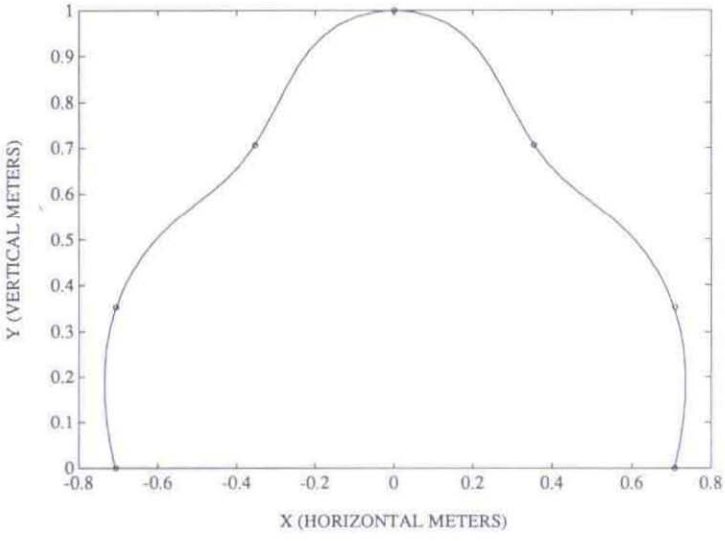

Figure 2. Cartesian path for nominal trignometric spline.

Hildreth's algorithm was used for the minimum jerk trajectory with nonzero knot tolerances (as described in Section 3.3.1). Note that Hildreth's algorithm is iterative in nature. So, it could theoretically take an infinite number of iterations before convergence is achieved. Therefore, some error y_e in the solution is allowed. Once Hildreth's algorithm achieves a solution with knot errors within $\pm(y_{toli} + y_e)$, the algorithm is considered to have converged. The allowable error y_e was chosen to be 0.1°. So the actual allowable knot tolerances were 4.1°, but the parameters y_{toli} were fixed at 4°.

For the minimum torque trajectories, Powell's method of nonlinear parameter optimization (as described in Section 3.2) was implemented on a DEC Vax 8820 running VMS 5.1. Powell's method was used to perform $(n - 1)$ separate $3P$-dimensional minimization problems, as indicated in eq. (26). The number of knots is $(n + 1)$, and the number of joints is P (two for the manipulator considered in this section). The weighting matrix R of eq. (21) was taken to be the identity matrix. The algorithm was considered to have converged when the objective function decreased by less than 0.5%. The additional minimization with respect to the knot angles (for the case of minimum torque with nonzero knot tolerances) was considered to have converged when the knot angle under consideration changed by less than 0.5°.

The resulting Cartesian space trajectories are shown in Figures 2–6. Note the strange motion of the minimum torque spline in Figure 5. This is apparently due to the singularity at $(x,y) = (0,1)$ and points out the fact that minimum torque trajectories take the dynamics of the robot into account. Therefore, the resulting trajectory may not agree with intuition. Also, note that gravity increases the required torque when the end-effector is rising but decreases the

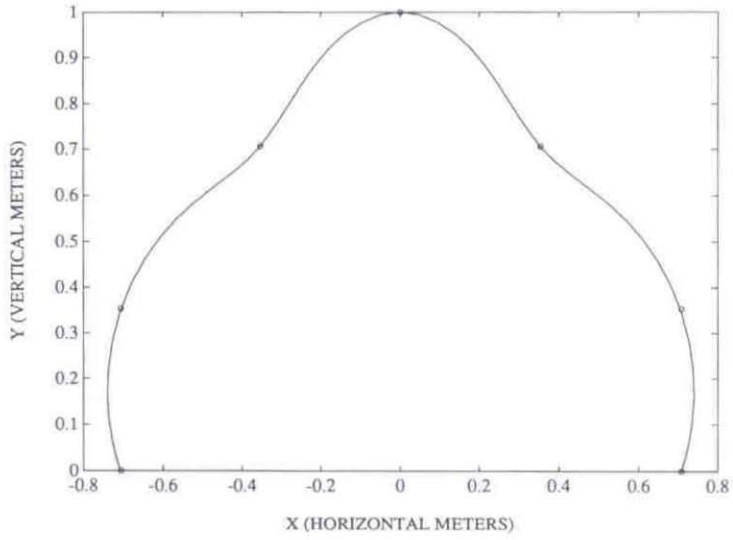

Figure 3. Cartesian path for minimum jerk spline.

required torque when the end-effector is descending. Therefore, the minimum torque trajectory may not be symmetric, as seen in Figure 5.

A comparison of the various objective functions is given in Table II. The numbers in Table II are rad^2/s^5 for the jerk objective function and $(Newton\text{-}meter)^2 \cdot s$ for the torque objective function. Note from Table II the sizeable

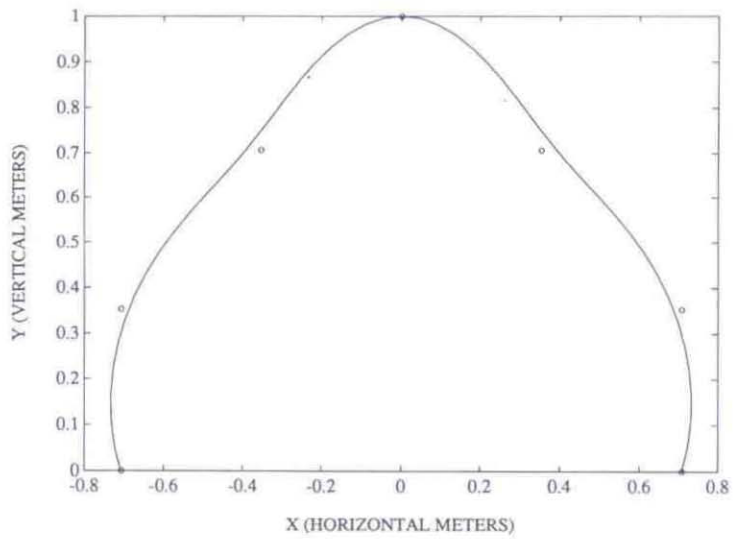

Figure 4. Cartesian path for minimum jerk (nonzero knot tolerance) spline.

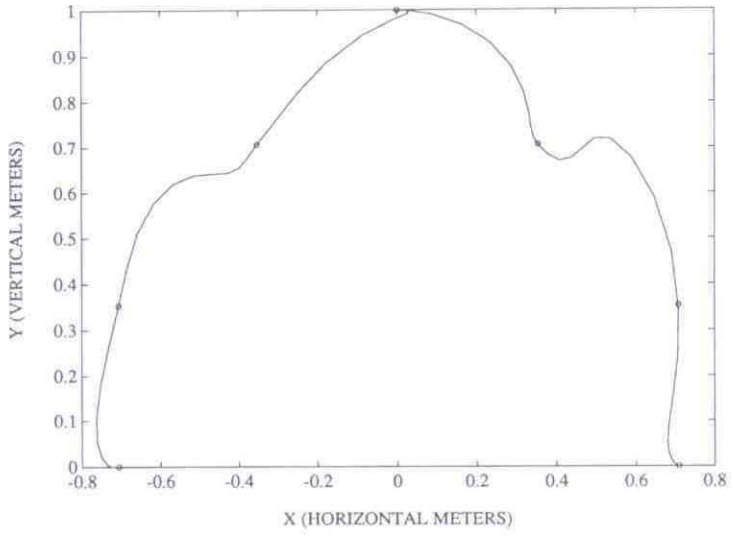

Figure 5. Cartesian path for minimum torque spline.

improvement in the torque objective function when optimization is used. Even the minimum jerk splines decrease the torque requirement by a factor of five or six when compared to the nominal splines. This indicates that the minimization of jerk is a big step toward the minimization of torque. In contrast, the use of minimum torque splines does not result in any improvement of the jerk objective function when compared to the nominal splines.

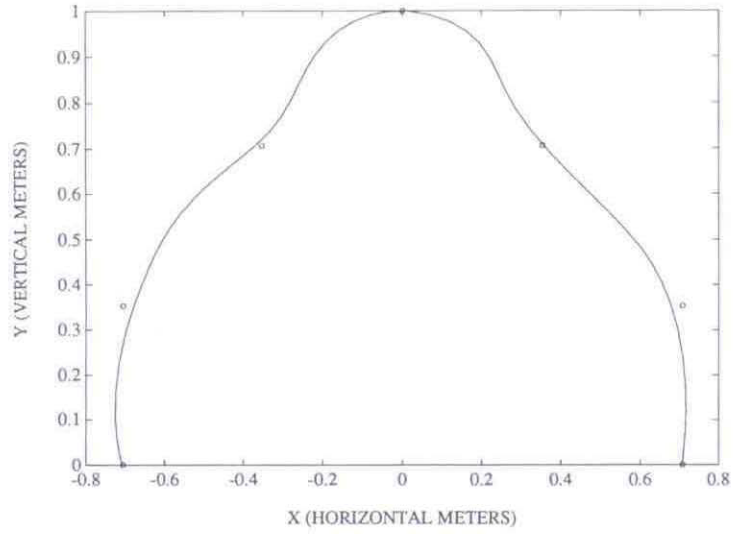

Figure 6. Cartesian path for minimum torque (nonzero knot tolerance) spline.

Table II. Objective functions for various trigonometric splines.

	Type of trigonometric spline				
		Minimum jerk		Minimum torque	
Objective function	Nominal	Zero knot tolerance	Nonzero knot tolerance	Zero knot tolerance	Nonzero knot tolerance
Joint 1 jerk	1.453	0.2229	0.1741	1.608	2.355
Joint 2 jerk	2.351	0.8190	0.4028	2.342	1.278
Torque	27,341	3674	3021	557	507

Table II shows that the introduction of nonzero knot tolerances results in a decrease of the objective function under consideration. This is as expected. The optimization algorithm is given more free parameters, and this results in better performance.

Table III shows the computational effort that was required for each spline. The nominal spline and minimum jerk splines have closed-form solutions, and so their computational effort can be measured in *flops* (floating point operations). The other splines in Table III require iterative solutions, and so their computational effort is measured in CPU time on a Vax 8820 computer. It has been shown that the computational effort increases linearly with the number of knots. See Simon and Isik for details.[9,10]

5. CONCLUSION

A general trigonometric spline robot trajectory formulation algorithm has been summarized. The input to the algorithm is a sequence of $(n + 1)$ joint angles for each joint, which are determined by performing inverse kinematics on a sequence of Cartesian knots.

It has been shown that the use of trigonometric splines for robot path planning is very amenable to path optimization subject to user-specified knot tolerances. The knots may be chosen to avoid obstacles. So, the robot path does not need to path exactly through the knots but rather *near* the knots. This possibility makes optimization subject to user-specified knot tolerances a desirable feature of a path planning method. The objective function under consideration can decrease significantly if the knot tolerances are used wisely. The optimization procedures presented in this article are iterative and thus cannot be performed

Table III. Vax 8820 computational effort.

Type of spline	Computational effort
Nominal	760 flops
Minimum jerk	767 flops
Minimum jerk with knot tolerances	2.7 s
Minimum torque	55 s
Minimum torque with knot tolerances	390 s

in real time. But, if the objective function is minimum jerk subject to knot tolerances, then the optimization problem reduces to a quadratic programming problem with linear constraints. This is a well-known problem, and there are several ways of solving it.

If the objective function includes the dynamics of the robot (e.g., torque), then the optimization problem must be solved using an iterative parameter optimization method. This is due to the nonlinearity and decoupled nature of robot dynamics. There is presently no known theory for closed-form minimization of arbitrary, nonlinear functions. But the decoupled nature of trigonometric splines can be exploited. The decoupling of the spline segments means that the optimization problem can be split into many smaller subproblems (one for each knot). This decreases the computational effort by a significant amount. The simulation results of this article indicate that the minimization of a torque objective function can result in a decrease of torque by a factor of 25 or more. This would result in less wear and tear on the robot and lower power requirements. Both of these results would be attractive to robot users.

The authors are grateful to the reviewers for their thoughtful suggestions for improvements to this article.

References

1. J. Craig, *Introduction to Robotics*, Addison-Wesley, Reading, MA, 1989.
2. S. Chand and K. Doty, "On-line polynomial trajectories for robot manipulators," *International J. of Robotics Research*, **4,** 38–48 (1985).
3. S. Thompson and R. Patel, "Formulation of joint trajectories for industrial robots using B-splines," *IEEE Trans. on Industrial Electronics*, **34,** 192–199 (1987).
4. C. Lin, P. Chang, and J. Y. S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Trans. on Automatic Control*, **28,** 1066–1074 (1983).
5. C. Lin and P. Chang, "Joint trajectories of mechanical manipulators for Cartesian path approximation," *IEEE Trans. on Systems, Man, and Cybernetics*, **13,** 1094–1102 (1983).
6. J. Y. S. Luh and C. Lin, "Approximate joint trajectories for control of industrial robots along Cartesian paths," *IEEE Trans. on Systems, Man, and Cybernetics*, **14,** 444–450 (1984).
7. D. Simon and C. Isik, "Optimal trigonometric robot joint trajectories," *Robotica*, **9,** 379–386 (1991).
8. D. Simon and C. Isik, "The generation and optimization of trigonometric joint trajectories for robotic manipulators," *Proc. of American Control Conf.* Vol. 2, 1991, pp. 2027–2032.
9. D. Simon, "*A Unified Approach to Robot Path Planning Using Trigonometric Splines*," Ph.D. dissertation, Syracuse University, Syracuse, NY, 1991.
10. D. Simon and C. Isik, "Computational complexity and path error analyses of trigonometric joint trajectories," *International J. of Robotics and Automation*, **7,** 179–185 (1992).
11. I. Schoenberg, "On trigonometric spline interpolation," *J. of Mathematics and Mechanics*, **13,** 795–825 (1964).
12. P. Koch, "Error bounds for interpolation by fourth order trigonometric splines," in *Approximation Theory and Spline Functions*, S. Singh, J. Burry, and B. Watson, Eds., D. Reidel, Dordrecht, Holland, 1984, pp. 349–360.

13. T. Lyche and R. Winther, "A stable recurrence relation for trigonometric B-splines," *J. of Approximation Theory*, **25,** 266–279 (1979).
14. P. Koch and T. Lyche, "Bounds for the error in trigonometric hermite interpolation," in *Quantitative Approximation*, R. Devore and K. Scherer, Eds., Academic Press, New York, 1980, pp. 185–196.
15. Y. Chen and M. Vidyasagar, "Optimal control of robotic manipulators in the presence of obstacles," *J. of Robotic Systems*, **7,** 721–740 (1990).
16. B. Heimann, H. Loose, and G. Schuster, "Contribution to optimal control of industrial robots," *Proc. of 4th CISM-IFToMM Symp. on Theory and Practice of Robots and Manipulators*, 1981, pp. 162–169.
17. C. Neuman and A. Sen, "A suboptimal control algorithm for constrained problems using cubic splines," *Automatica*, **9,** 601–613 (1973).
18. D. Schmitt, A. Soni, V. Srinivasam, and G. Naganathan, "Optimal motion programming of robot manipulators," *J. of Mechanisms, Transmissions, and Automation in Design*, **107,** 239–244 (1985).
19. J. Vlassenbroeck and R. Van Dooren, "A Chebyshev technique for solving nonlinear optimal control problems," *IEEE Trans. on Automatic Control*, **33,** 333–340 (1988).
20. V. Yen and M. Nagurka, "Optimal trajectory planning of robotic manipulators via quasi-linearization and state parameterization," *Proc. of IEEE International Conf. on Robotics and Automation*, Vol. 2, 1989, pp. 1116–1121.
21. V. Yen and M. Nagurka, "Fourier-based optimal control approach for structural systems," *J. of Guidance, Control, and Dynamics*, **13,** 265–276 (1990).
22. K. Kyriakopoulos and G. Saridis, "Minimum jerk path generation," *Proc. of IEEE International Conf. on Robotics and Automation*, Vol. 1, 1988, pp. 364–369.
23. J. Flanagan and D. Ostry, "Trajectories of human multi-joint arm movements: Evidence of joint level planning," in *Experimental Robotics I, The First International Symposium*, V. Hayward and O. Khatib, Eds., Springer-Verlag, New York, 1990, pp. 594–613.
24. W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes*, Cambridge University Press, New York, 1986.
25. R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, Cambridge, MA, 1981.
26. C. Hildreth, "A quadratic programming procedure," *Naval Research Logistics Quarterly*, **4,** 79–85 (1957).
27. D. Luenberger, *Optimization by Vector Space Methods*, Wiley, New York, 1968.